# **React**

**VS Code Shortcuts:**

Shift + Alt + O Remove unused imports

Shift + Alt + ⬇ Duplicate current line

Ctrl + / Comment current line

Shift + Alt + F Formatting

Alt + ⬆ Move line up

Alt + ⬇ Move line down

Ctrl + D Select next word

Ctrl + + Zoom In

Ctrl + - Zoom Out

**ריאקט**

Framework מבית Facebook לבניית אתר Frontend מבוסס קומפוננטות בשיטת SPA.

**SPA**

Single Page Application

זהו אתר המכיל דף html אחד ויחיד. כלל ה-UI כבר קיים בצד הלקוח ב-JS. תפריט רק "מזריק" את הדף המתאים ל-html האחד והיחיד (לרוב נקרא index.html).

**Component – קומפוננטה**

חלק קטן מה-UI.

**TypeScript**

זו שפה שפותחה ע"י מייקרוסופט ב-2013 שהופכת את עצמה ל-JavaScript.

**התקנות:**

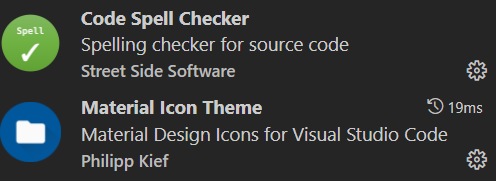
התקנת React פעם אחת במכונה:

npm i -g create-react-app

יצירת פרויקט חדש (בתיקייה כלשהי):

create-react-app games-website --template typescript

התקנות תוספים ל-VS Code:



**Directory Structure**

src

Assets

Images

MyImage.jpg

Models

ProductModel.ts

SupplierModel.ts

Services

ProductService.ts

SupplierService.ts

Utils

AppConfig.ts

ImageUtil.ts

Components

ProductArea

ProductList

ProductList.css

ProductList.tsx

ProductList.test.tsx

ProductCard

ProductCard.css

ProductCard.tsx

SupplierArea

SupplierCard

SupplierCard.css

SupplierCard.tsx

**CSS-Leaking**

זהו מצב בו CSS של קומפוננטה אחת משפיע על עיצוב של קומפוננטות אחרות.

**Interpolation**

הכנסת ערך של משתנה, ערך מוחזר מפונקציה וכדומה לתוך ה-HTML של הקומפוננטה.

**CSS-Module**

יצירת קובץ CSS המסתיים ב- .module.css

זה מאפשר לבנות בתוכו Class-ים ולייבא קובץ זה לתוך אובייקט ואז להשתמש ב-Class-ים השייכים לאובייקט זה בקומפוננטה.

ריאקט תמציא שם רנדומלי וייחודי לכל Class.

ספרייה ליצירת קומפוננטות בריאקט:

npm i -g react-cli-snippets

יצירת קומפוננטה:

create fc KittenArea/Kitten

יצירת קומפוננטה בשיטת CSS-Module:

create fc PuppyArea/Puppy --module

**Conditional Rendering**

זהו מצב בו אנו מציגים קומפוננטה או חלק ממנה כתלות בתנאי מסוים.

**React Hooks**

אלו פונקציות מיוחדות המתחילות בתחילית use ומבצעות פעולה מסוימת אך ורק ב-Functional Components או ב-Custom Hooks אחרים.

1. חייבות להתחיל ב-use.
2. ניתנות לשימוש רק ב-FC.
3. אפשר לקרוא להן אך ורק ישירות בתוך ה-FC, לא בתוך פונקציות פנימיות.
4. אי אפשר לקרוא להן בתוך תנאים או לולאות.

**State**

המידע המשויך לקומפוננטה כך שאם הוא משתנה ברצוננו להציג את הקומפוננטה מחדש.

פונקציה המאפשרת לבצע זאת נקראת useState.

**Side-Effect**

זהו מצב של פעולה שקומפוננטה מבצעת שגורמת לשינוי כלשהו או פעילות כלשהי מחוץ לגבולות הקומפוננטה.

לדוגמה, קומפוננטה גולשת לשרת מרוחק.

לדוגמה, קומפוננטה שהפעילה טיימר.

אסור לבצע Side-Effects ישירות בתוך פונקציית הקומפוננטה!!!

אם ברצוננו לבצע Side-Effect כאשר קומפוננטה עולה, יש לבצע זאת ב-Hook בשם useEffect – מבצע את הקוד בתוכו פעם אחת או כאשר מידע מסוים שנגדיר ישתנה.

**Component Lifecycle**

מחזור חיים של קומפוננטה.

קומפוננטה מתחילה לחיות ברגע שהיא מוצגת. ריאקט מחברת אותה ל-Virtual DOM. ברגע שריאקט סיימה לחבר את הקומפוננטה, היא מפעילה את useEffect. זה נקרא Mounted.

ברגע שקומפוננטה לא מוצגת יותר על המסך, היא עוברת תהליך שנקרא Unmount בו ריאקט מפרקת את הזכרון, והורסת את הקומפוננטה. אם קומפוננטה משתמשת במשאבים שגוזלים זמן, זכרון וכדומה, יש להפסיק את השימוש במשאבים אלו כאשר מתבצע Unmount.

תרגיל:

הורידו מגוגל 5 תמונות של משחקים ל-Assets/Images.

צרו קומפוננטה בשם RandomGames הנמצאת באזור GameArea.

על הקומפוננטה להציג בכל 3 שניות תמונה רנדומלית מחמשת התמונות הנ"ל.

הוסיפו עיצוב אסתטי.

יש להתחיל טיימר ב-useEffect, יש לדאוג לסגירת הטיימר ב-Unmount וכדומה.

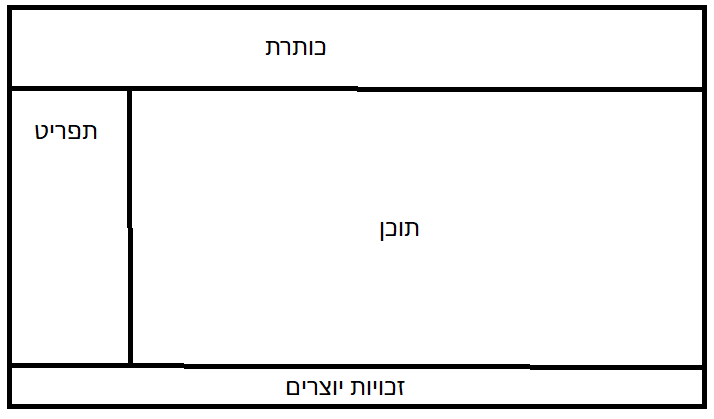
**Layout**

הסידור הכללי של האתר.

ה-Layout נבנה ע"י קומפוננטה שהינה הראשונה שמוצגת ב-index.html.

שאר הקומפוננטות נמצאות בתוך ה-Layout.

לדוגמה:



**Routing**

זהו מנגנון המאפשר להציג קומפוננטה כתלות בנתיב המופיע בכתובת הדפדפן (נקרא Route).

בריאקט עלינו להתקין ספרייה בתוך הפרויקט המכילה כלים עבור Routing:

npm i react-router-dom @types/react-router-dom

**Lazy Loading**

זהו מצב בו המערכת טוענת משהו רק כאשר הוא נדרש לשימוש.

בעולם ה-Routing, ניתן לטעון דף מהשרת רק אם המשתמש גולש אליו. זה גורם לטעינה הראשונית של האתר להיות מהירה יותר.

**REST API**

שירות שחושף מידע

REST: REpresentational State Transfer

שירות REST API של Northwind ניתן להתקנה מקומית ע"י:

npm i -g northwind-rest-api

הרצת ה-REST API:

northwind

אנו גולשים לשרת מרוחק בריאקט ע"י ספרייה מקומית שיש להתקין בפרויקט:

npm i axios

**Service**

מחלקה המכילה קוד לוגי בלבד. לא קשורה ל-UI.

**Promise**

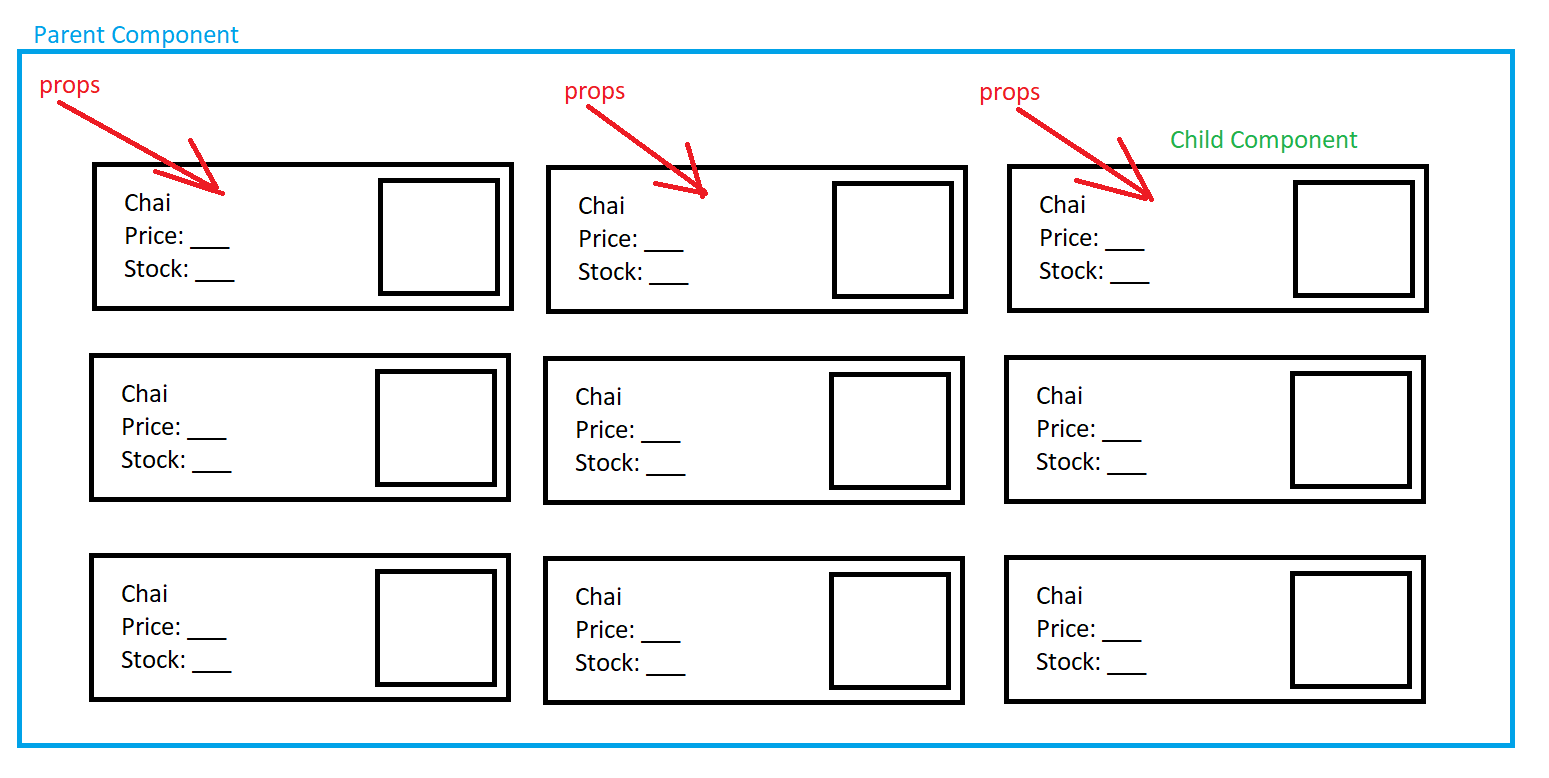
אובייקט Native ב-JavaScript המיועד להריץ בתוכו קוד אסינכרוני ולדווח הצלחה או כישלון.

await

1. ממתינה.
2. לא תוקעת את ה-Call Stack כי ההמתנה הינה ב-Web APIs
3. בסיום הפעילות מחלצת את המידע שה-Promise מדווח ומחזירה אותו.

**props**

זהו אובייקט שניתן לקבל לפונקציית ה-Component שה-Parent Component שלנו יכול לשלוח.



**Route Parameter**

זהו מידע שניתן לשלוח כחלק מה-Route המוצג ב-URL. לדוגמה:

/products/details/12

הקוד יכול להשתנות ולכן הוא לא מחרוזת קבועה אלא פרמטר שמשולב ב-Route.

**Forms**

קליטת מידע מתיבה אחת בריאקט:

const [price, setPrice] = useState<number>("");

function handleChange(args: SyntheticEvent): void {

const value = +(args.target as HTMLInputElement).value;

setPrice(value)

}

<input type="number" onChange={handleChange} value={price}>

קיימת ספרייה פופולרית לניהול טפסים במקום לכתוב את הקוד הנ"ל עבור כל תיבה.

התקנה:

npm i react-hook-form

ספרייה להצגת הודעות:

npm i notyf

תרגול כיתה:

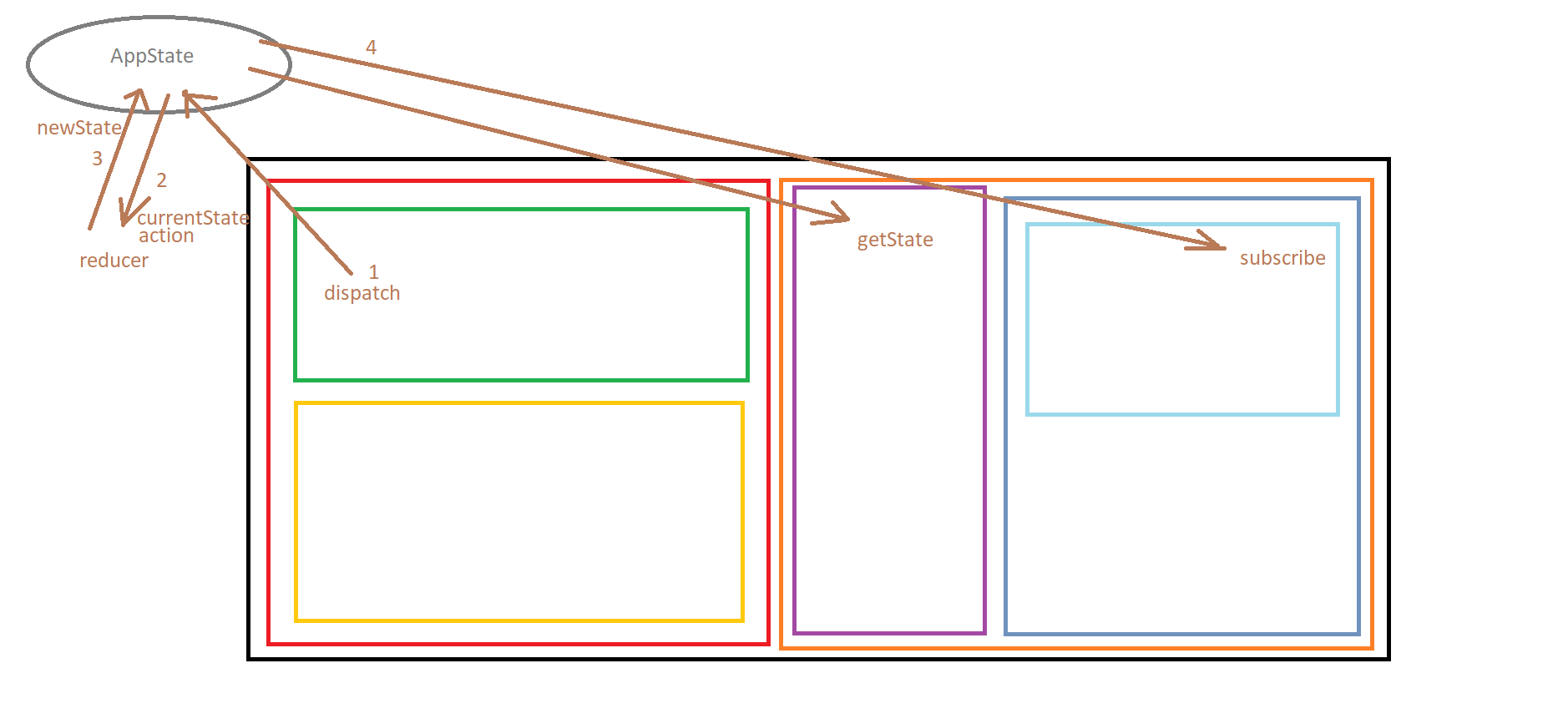
עליכם לבנות דף להצגת עובדי חברת Northwind בטבלה. לצורך כך יש לבצע את השלבים הבאים:

1. הגדרת הכתובת [http://localhost:3030/api/employees/](http://localhost:3030/api/employees//) ב-AppConfig.
2. בניית מודל EmployeeModel הכולל בתוכו את המידע המוחזר עבור כל עובד.
3. בניית EmployeeService המכיל פונקציה להחזרת העובדים.
4. בניית קומפוננטה להצגת טבלת העובדים.
5. בניית Route להצגת הקומפוננטה ותפריט להצגת ה-Route הזה.
6. הצגת טבלת העובדים הכולל את כלל המאפיינים, כולל תמונה.

**Redux**

זו ספרייה לניהול State ברמת האפליקציה.

אם קיים לנו מידע הנדרש במספר קומפוננטות קיימת בעיה לשתף את המידע הזה בין הקומפוננטות השונות.



מונחי Redux:

1. AppState: (נקרא גם Global State) כלל המידע הקיים ברמת האפליקציה.

לדוגמה, אפליקציה מנהלת מוצרים, עובדים וספקים. לכן כלל המידע יהיה אוסף מוצרים, אוסף עובדים ואוסף ספקים.

1. Slice: חלק אחד בכלל המידע. לרוב זה אובייקט אחד או אוסף אחד.

לדוגמה, רק אוסף המוצרים.

1. Reducer: זו פונקציה שאנו בונים ונותנים אותה ל-Redux (רושמים אותה). פונקציה זו מבצעת שינוי אחד על המידע (על Slice). אנו לא קוראים ל-Reducers בעצמנו, אלא Redux מפעילה אותם.

כל פונקציית Reducer תמיד מקבלת את הפרמטרים הבאים:

פרמטר ראשון: ה-State הנוכחי של ה-Slice.

פרמטר שני: אובייקט ה-Action ש-Dispatch שלחה לנו.

על הפונקציה לבצע את הפעולה הדרושה ולהחזיר State חדש. אסור לה לבצע את הפעולה על ה-State שקיבלה.

לדוגמה, הוספת מוצר בודד לאוסף המוצרים.

1. Action: אובייקט המתאר פעולה אחת על המידע. כל Reducer מקבל Action אחד לביצוע.

ה-Action מכיל שני מאפיינים: א. האובייקט עצמו המכיל את המידע (נקרא payload). ב. מחרוזת המתארת מה לבצע – לשימוש פנימי של Redux (נקרא type).

לדוגמה, אם ברצוננו להוסיף מוצר, יש אובייקט המכיל את המוצר החדש + מחרוזת המתארת שיש להוסיף מוצר. אובייקט זה הינו ה-Action.

1. Action Creator: פונקציה היוצרת אובייקט Action מוכן לשימוש.
2. Dispatch: פונקציה של Redux שאנו קוראים לה, שולחים לה Action והיא מפעילה את ה-Reducer המתאים.

לדוגמה קריאה ל-dispatch ושליחת Action עבור הוספת מוצר.

1. Store: האובייקט הראשי של Redux המנהל את כל הפעילות.
2. Middleware: פונקציה שאנו יכולים לבנות המבצעת פעולה כלשהי בכל פעם שקוראים Dispatch.

התקנות:

npm i @reduxjs/toolkit

ספרייה המכילה Hooks שימושיים:

npm i react-redux @types/react-redux

# תרגול

1. צרו את הקוד הדרוש עבור העלאת עובדי Northwind לרמת ה-Global State:
2. Reducers הכוללים init, add, update, delete.
3. employeesSlice
4. הוספת אוסף העובדים ל-AppState
5. ב-EmployeesService, יש לבדוק האם העובדים כבר קיימים ב-Global State, אם לא – להביא מהשרת ולהוסיף ל-Global State.
6. בלחיצה על תמונת עובד מהטבלה – לעבור לקומפוננטה המציגה רק את העובד הזה. יש להביא את העובד מה-Global State. אם לא נמצא שם – להביא מהשרת.
7. הוספת קומפוננטה להוספת עובד חדש השולחת את העובד לשרת + מעדכנת את ה-Global State.
8. הוספת לחצן בטבלה לביצוע מחיקה של עובד חדש מהשרת + מה-Global State.
9. אתגר: קומפוננטה לעדכון עובד קיים.

Auth – Authentication & Authorization

Authentication: אימות המשתמש (הרשמה/כניסה/יציאה – Register/Login/Logout)

Authorization: הרשאות – מה המשתמש יכול לבצע. מוגדר לפי תפקידים – Roles.

JWT: Json Web Token

שיטת Authentication שבה השרת שולח ללקוח מחרוזת המייצגת את ההתחברות שלו. הלקוח חייב לשמור מחרוזת זו ולשלוח אותה לשרת בכל פעם שהוא רוצה לבצע פעולה הדרושה הרשאה כלשהי.

ספרייה לחילוץ האובייקט הקיים ב-Token:

npm i jwt-decode

**Interceptor**

זהו קוד שניתן לרשום שרץ אוטומטית בכל Request או בכל Response.

הוא יכול לשנות/להוסיף מידע לאובייקט ה-Request שנשלח או לאובייקט ה-Response שמתקבל.

MUI – Material UI

זו ספרייה פופולרית המכילה רכיבים מעוצבים בריאקט.

מכילה בנוסף אנימציות על פעולות ברכיבים אלו.

אתר רשמי:

<https://mui.com>

<https://mui.com/material-ui/material-icons>

התקנה:

npm i @mui/material @emotion/react @emotion/styled @mui/icons-material

**Firebase**

זהו Cloud של גוגל. מיועד להכיל פרויקט Frontend. הוא מספק Backend (BaaS).

התקנה גלובלית המאפשרת להעלות פרויקט שבנינו ל-Firebase:

npm i -g firebase-tools

# בהצלחה!